**OPERATING SYSTEM**

**CYCLE-1**

**1a. FCFS**

**Code:**

#include <stdio.h>

void bubblesort(int n, int process[], int arrival[], int brust[]);

int main() {

int n;

printf("Enter number of processes: ");

scanf("%d", &n);

int process[n];

int arrival[n];

int brust[n];

for (int i = 0; i < n; i++) {

process[i] = i + 1;

}

printf("Enter arrival times: ");

for (int i = 0; i < n; i++) {

scanf("%d", &arrival[i]);

}

printf("Enter burst times: ");

for (int i = 0; i < n; i++) {

scanf("%d", &brust[i]);

}

bubblesort(n, process, arrival, brust);

int ct[n], tt[n], wt[n];

int total\_time = 0;

for (int i = 0; i < n; i++) {

if (total\_time < arrival[i]) {

total\_time = arrival[i];

}

ct[i] = total\_time + brust[i];

total\_time = ct[i];

}

for (int i = 0; i < n; i++) {

tt[i] = ct[i] - arrival[i];

wt[i] = tt[i] - brust[i];

}

printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

for (int i = 0; i < n; i++) {

printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", process[i], arrival[i], brust[i], ct[i], tt[i], wt[i]);

}

return 0;

}

void bubblesort(int n, int process[], int arrival[], int brust[]) {

int temp;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

if (arrival[j] > arrival[j + 1]) {

temp = arrival[j];

arrival[j] = arrival[j + 1];

arrival[j + 1] = temp;

temp = brust[j];

brust[j] = brust[j + 1];

brust[j + 1] = temp;

temp = process[j];

process[j] = process[j + 1];

process[j + 1] = temp;

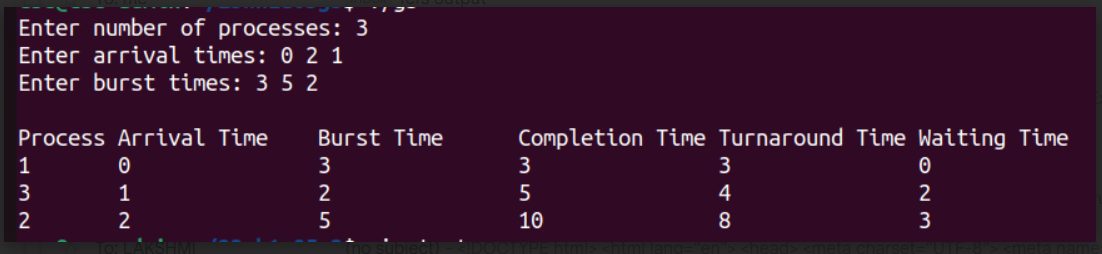
}

}

}

}

**Output:**

****

**1b. SJF**

**Code:**

#include<stdio.h>

void bubblesort(int n, int process[], int arrival[], int brust[]);

int main()

{

int n;

printf("enter number of processes: ");

scanf("%d",&n);

int process[n];

int arrival[n];

int brust[n];

for(int i=0; i<n; i++)

{

process[i] = i+1;

}

printf("enter arrival times: ");

for(int i=0; i<n; i++)

{

scanf("%d",&arrival[i]);

}

printf("enter brust times: ");

for(int i=0; i<n; i++)

{

scanf("%d",&brust[i]);

}

bubblesort(n, process, arrival, brust);

int ct[n], wt[n], tt[n];

int total\_time = 0;

for (int i = 0; i < n; i++) {

if (total\_time < arrival[i])

{

total\_time = arrival[i];

}

ct[i] = total\_time + brust[i];

total\_time = ct[i];

}

for (int i = 0; i < n; i++) {

tt[i] = ct[i] - arrival[i];

wt[i] = tt[i] - brust[i];

}

printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

for (int i = 0; i < n; i++) {

printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", process[i], arrival[i], brust[i], ct[i], tt[i], wt[i]);

}

return 0;

}

void bubblesort(int n, int process[], int arrival[], int brust[])

{

int temp;

for(int i=0; i<n-1; i++)

{

for(int j=0; j<n-1; j++)

{

if(brust[j]>brust[j+1])

{

temp = arrival[j];

arrival[j] = arrival[j+1];

arrival[j+1] = temp;

temp = brust[i];

brust[i] = brust[i+1];

brust[i+1] = temp;

temp = process[i];

process[i] = process[i+1];

process[i+1] = temp;

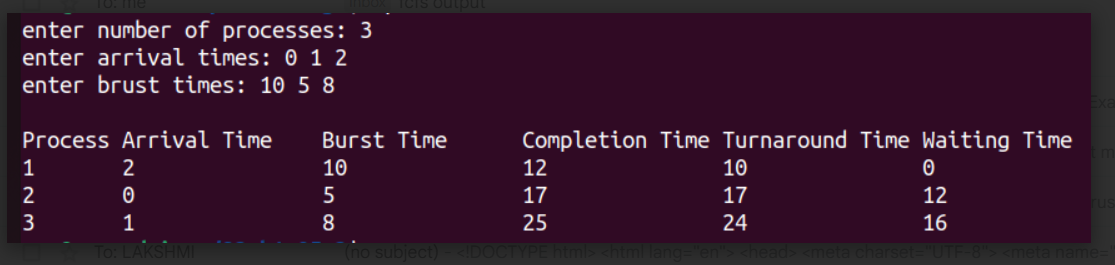
}

}

}

}

**Output:**



**1c. Round Robin**

**Code:**

#include<stdio.h>

void main() {

int n, tq;

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter the time quantum: ");

scanf("%d", &tq);

int burst[n], arrival[n], process[n];

for (int i = 0; i < n; i++) {

process[i] = i + 1;

}

printf("Give burst time for each process: ");

for (int i = 0; i < n; i++) {

scanf("%d", &burst[i]);

}

printf("Give arrival time for each process: ");

for (int i = 0; i < n; i++) {

scanf("%d", &arrival[i]);

}

int wt[n], rt[n], current\_t[n], ct[n], tt[n], total\_time = 0;

for (int i = 0; i < n; i++) {

wt[i] = 0;

rt[i] = burst[i];

current\_t[i] = 0;

}

int completed = 0;

while (completed < n) {

for (int i = 0; i < n; i++) {

if (rt[i] > 0 && arrival[i] <= total\_time) {

if (rt[i] > tq) {

total\_time += tq;

rt[i] -= tq;

} else {

total\_time += rt[i];

ct[i] = total\_time;

rt[i] = 0;

completed++;

}

}

}

}

for (int i = 0; i < n; i++) {

tt[i] = ct[i] - arrival[i];

wt[i] = tt[i] - burst[i];

}

printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");

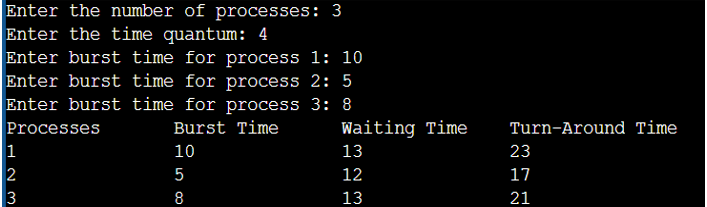
for (int i = 0; i < n; i++) {

printf("%d\t%d\t%d\t%d\n", process[i], burst[i], wt[i], tt[i]);

}

}

**Output:**



**1d. Priority**

**Code:**

#include <stdio.h>

void calculateWaitTime(int count, int arrival[], int burst[], int wait[]) {

int currentTime = 0;

for (int i = 0; i < count; i++) {

if (currentTime < arrival[i]) {

currentTime = arrival[i];

}

wait[i] = currentTime - arrival[i];

if (wait[i] < 0) {

wait[i] = 0;

}

currentTime += burst[i];

}

}

void calculateTurnaroundTime(int count, int burst[], int wait[], int turnaround[]) {

for (int i = 0; i < count; i++) {

turnaround[i] = wait[i] + burst[i];

}

}

void sortProcesses(int count, int proc[], int arrival[], int burst[], int priority[]) {

for (int i = 0; i < count - 1; i++) {

for (int j = i + 1; j < count; j++) {

if (priority[i] > priority[j] || (priority[i] == priority[j] && arrival[i] > arrival[j])) {

int temp = priority[i];

priority[i] = priority[j];

priority[j] = temp;

temp = arrival[i];

arrival[i] = arrival[j];

arrival[j] = temp;

temp = burst[i];

burst[i] = burst[j];

burst[j] = temp;

temp = proc[i];

proc[i] = proc[j];

proc[j] = temp;

}

}

}

}

void displayResults(int count, int proc[], int arrival[], int burst[], int priority[], int wait[], int turnaround[]) {

printf("\nProcess\tArrival Time\tPriority\tBurst Time\tWaiting Time\tTurnaround Time\n");

for (int idx = 0; idx < count; idx++) {

printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[idx], arrival[idx], priority[idx], burst[idx], wait[idx], turnaround[idx]);

}

}

void displayAverages(int count, int wait[], int turnaround[]) {

double totalWait = 0, totalTurnaround = 0;

for (int idx = 0; idx < count; idx++) {

totalWait += wait[idx];

totalTurnaround += turnaround[idx];

}

printf("\nAverage Waiting Time: %.2lf\n", totalWait / count);

printf("Average Turnaround Time: %.2lf\n", totalTurnaround / count);

}

int main() {

int numProcesses;

printf("Enter the number of processes: ");

scanf("%d", &numProcesses);

int processIDs[numProcesses], arrivalTimes[numProcesses], burstTimes[numProcesses], priorities[numProcesses];

int waitingTimes[numProcesses], turnaroundTimes[numProcesses];

for (int idx = 0; idx < numProcesses; idx++) {

processIDs[idx] = idx + 1;

printf("Enter arrival time for process P%d: ", processIDs[idx]);

scanf("%d", &arrivalTimes[idx]);

printf("Enter burst time for process P%d: ", processIDs[idx]);

scanf("%d", &burstTimes[idx]);

printf("Enter priority for process P%d: ", processIDs[idx]);

scanf("%d", &priorities[idx]);

}

sortProcesses(numProcesses, processIDs, arrivalTimes, burstTimes, priorities);

calculateWaitTime(numProcesses, arrivalTimes, burstTimes, waitingTimes);

calculateTurnaroundTime(numProcesses, burstTimes, waitingTimes, turnaroundTimes);

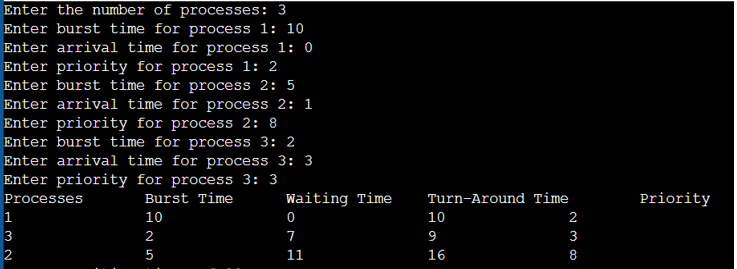
displayResults(numProcesses, processIDs, arrivalTimes, burstTimes, priorities, waitingTimes, turnaroundTimes);

displayAverages(numProcesses, waitingTimes, turnaroundTimes);

return 0;

}

**Output:**



**2a. Write a program to implement Process management system calls viz., fork, exit, wait, waitpid, exec.**

**Code:**

#include<stdio.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/types.h>

int main()

{

pid\_t pid = fork();

if(pid < 0)

{

perror("process not created");

return 1;

}

else if(pid == 0)

{

printf("chid process is created with (pid:%d)\n",getpid());

execl("./g3","./g3",NULL); //here g3 is name of the executable file of fcfs file

//execl("me","me",NULL);

//execl("/bin/ls","ls",NULL);

perror("error");

return 1;

}

else

{

printf("parent process is created with (pid:%d)\n",getpid());

int status;

wait(&status);

if(WIFEXITED(status))

{

printf("\nchild process ended with status: %d\n",WEXITSTATUS(status));

}

else

{

printf("not exited properly");

}

printf("parent process ended\n");

}

return 0;

}

**Output:**

